What is the next step?

- We are going to normalize.
  - 1st NF: no multivalued attributes.
  - 2nd NF: functional dependencies.
    - Either only one PK exists
    - No non-key attribute exist
    - Or all non-key attribs depend on the full set of PK attribs.
  - 3rd NF
    - No transitive dependencies: dependencies between NON KEY ATTRIBUTES
- We are going to implement.
  - mySQL software

Chapter 6

SQL

SQL Is:

- Structured Query Language
- The standard for relational database management systems (RDBMS)
- SQL-92 Standard -- Purpose:
  - Specify syntax/semantics for data definition and manipulation
  - Define data structures
  - Enable portability
  - Allow for later growth/enhancement to standard

Benefits of a Standardized Relational Language

- Reduced training costs
- Productivity
- Application portability
- Application longevity
- Reduced dependence on a single vendor
- Cross-system communication

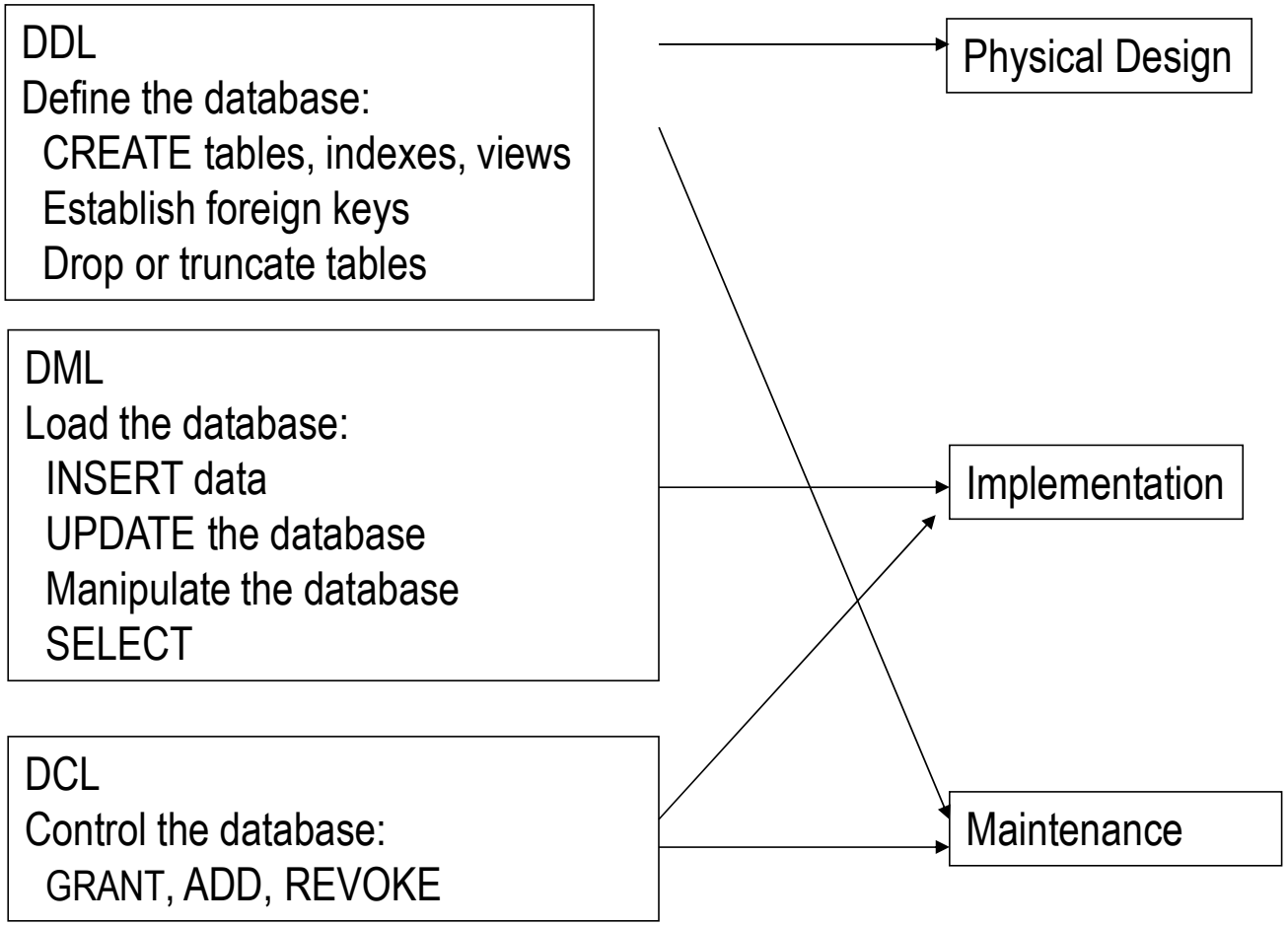
SQL Data types (from Oracle9i)

- String types
  - CHAR(n) – fixed-length character data, n characters long Maximum length = 2000 bytes
  - VARCHAR2(n) – variable length character data, maximum 4000 bytes
  - LONG – variable-length character data, up to 4GB. Maximum 1 per table
- Numeric types
  - NUMBER(p,q) – general purpose numeric data type
  - INTEGER(p) – signed integer, p digits wide
  - FLOAT(p) – floating point in scientific notation with p binary digits precision
- Date/time type
  - DATE – fixed-length date/time in dd-mm-yy form

SQL commands classified

- **Data Definition Language (DDL):**
  - Commands that define a database, including creating, altering, and dropping tables and establishing constraints.
- **Data Manipulation Language (DML)**
  - Commands that maintain and query a database.
- **Data Control Language (DCL)**
  - Commands that control a database, including administering privileges and committing data.

DDL, DML, DCL, and the database development process



DML

- Most frequently used Statements are DMLs
- Used for Populating tables with data
  - INSERT
- Querying tables
  - SELECT
  - MODIFY OR UPDATE
  - DELETE
  - MAKE TABLE
  - APPEND TABLE

Insert Statement

- Adds data to a table
- Inserting into a table
  - INSERT INTO CUSTOMER VALUES (001, ‘CONTEMPORARY Casuals’, 1355 S. Himes Blvd.’, ‘Gainesville’, ‘FL’, 32601);
- Inserting a record that has some null attributes requires identifying the fields that actually get data
  - INSERT INTO PRODUCT (PRODUCT_ID, PRODUCT_DESCRIPTION,PRODUCT_FINISH, STANDARD_PRICE, PRODUCT_ON_HAND) VALUES (1, ‘End Table’, ‘Cherry’, 175, 8);
- Inserting from another table
  - INSERT INTO CA_CUST SELECT * FROM CUSTOMER WHERE STATE = ‘CA’;

Update Statement

- Modifies data in existing rows

- UPDATE PRODUCT SET UNIT_PRICE = 775
  WHERE PRODUCT_ID = 7;

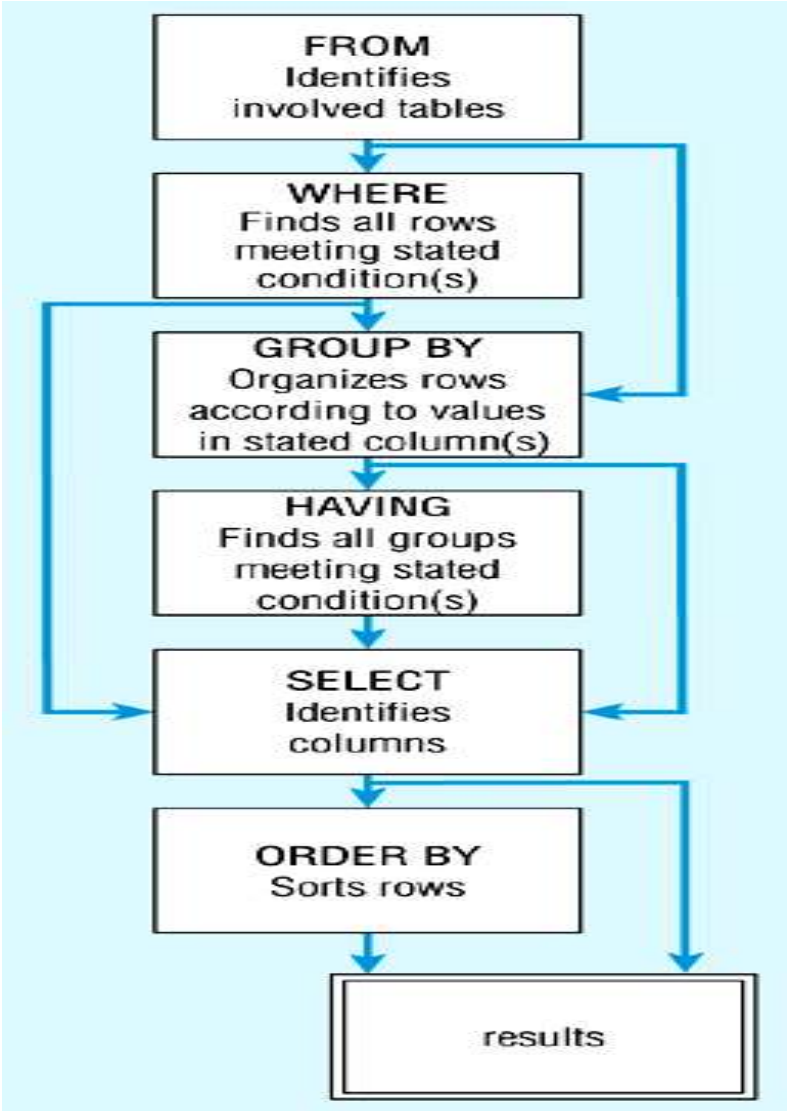
Delete Statement

- Removes rows from a table
- Delete certain rows
  - DELETE FROM CUSTOMER WHERE STATE = ‘HI’;
- Delete all rows
  - DELETE * FROM CUSTOMER;
  - DROP CUSTOMER;

The SELECT Statement

- Used for queries on single or multiple tables
- Clauses of the SELECT statement:
  - SELECT
    - List the columns (and expressions) that should be returned from the query
  - FROM
    - Indicate the table(s) or view(s) from which data will be obtained
  - WHERE
    - Indicate the conditions under which a row will be included in the result
  - GROUP BY
    - Indicate categorization of results
  - HAVING
    - Indicate the conditions under which a category (group) will be included
  - ORDER BY
    - Sorts the result according to specified criteria

SQL statement processing order (adapted from van der Lans, p.100)



SELECT Example

- Find products with standard price less than $275

- SELECT PRODUCT_NAME, STANDARD_PRICE
- FROM PRODUCT
- WHERE STANDARD_PRICE < 275

Comparison Operators in SQL

Table 7-3 Comparison Operators in SQL

Operator	*Meaning*
=	Equal to
>	Greater than
> =	Greater than or equal to
<	Less than
< =	Less than or equal to
<>	Not equal to
! =	Not equal to

SELECT Example – Boolean Operators

AND, OR, and NOT Operators for customizing conditions in WHERE clause

```
SELECT PRODUCT_DESCRIPTION,
PRODUCT_FINISH, STANDARD_PRICE
FROM PRODUCT
WHERE (PRODUCT_DESCRIPTION LIKE '%Desk'
OR PRODUCT_DESCRIPTION LIKE '%Table')
AND UNIT_PRICE > 300;
```

Note: the LIKE operator allows you to compare strings using wildcards. For example, the % wildcard in '%Desk' indicates that all strings that have any number of characters preceding the word "Desk" will be allowed

SELECT Example
Using a Function

- Using the COUNT ***aggregate function*** to find totals

- SELECT COUNT(*) FROM ORDER_LINE
  WHERE ORDER_ID = 1004;

End of dec4

SELECT Example with ALIAS

- Alias is an alternative column or table name

```
SELECT C.CUSTOMER_NAME AS
  NAME, C.CUSTOMER_ADDRESS
FROM CUSTOMER AS C
WHERE NAME = 'Home Furnishings';
```

SELECT Example –
Sorting Results with the ORDER BY Clause

- Sort the results first by STATE, and within a state by CUSTOMER_NAME

- SELECT CUSTOMER_NAME, CITY, STATE
- FROM CUSTOMER
- WHERE STATE IN ('FL', 'TX', 'CA', 'HI')
- ORDER BY STATE, CUSTOMER_NAME;

Note: the IN operator in this example allows you to include rows whose STATE value is either FL, TX, CA, or HI. It is more efficient than separate OR conditions

SELECT Example –
Categorizing Results Using the GROUP BY Clause

- For use with aggregate functions
  - ***Scalar aggregate***: single value returned from SQL query with aggregate function
  - ***Vector aggregate***: multiple values returned from SQL query with aggregate function (via GROUP BY)

```
SELECT STATE, COUNT(STATE)
FROM CUSTOMER
GROUP BY STATE;
```

Note: you can use single-value fields with aggregate functions if they are included in the GROUP BY clause

SELECT Example –
Qualifying Results by Categories Using the HAVING Clause

- For use with GROUP BY

```
SELECT STATE, COUNT(STATE)
FROM CUSTOMER
GROUP BY STATE
HAVING COUNT(STATE) > 1;
```

Like a WHERE clause, but it operates on groups (categories), not on individual rows. Here, only those groups with total numbers greater than 1 will be included in final result

Some DMLs

- **SELECTS**
  - **SELECT** Customer_Name, City
    FROM CUSTOMER
    WHERE State = ‘FL’;
  - **SELECT** *
    FROM PRODUCT
    WHERE Product_Finish = ‘Cherry’;
- **DELETES**
  - **DELETE** FROM CUSTOMER
    WHERE City = ‘BEVERLY HILLS’;

Some DMLs

- **UPDATES**
  - **UPDATE** CUSTOMER
    SET Customer_Name = ‘MARY SMITH’
    WHERE Customer_Name = ‘MARY JONES’;
  - **UPDATE** PRODUCT
    SET Unit_Price = 775
    WHERE Product_ID = 7;

Some more DMLs

- Creating new Table of selected records
  - **SELECT** * INTO WESTCUSTOMER
    FROM CUSTOMER
    WHERE CUSTOMER.State = ‘CA’;
- Appending to a Table
  - **INSERT INTO** WESTCUSTOMER
    SELECT *
    FROM CUSTOMER
    WHERE CUSTOMER.State = ‘HI’;

More DMLs

- BETWEEN Operator
  - **SELECT** Product_Description, Unit_Price
    FROM PRODUCT
    WHERE Unit_Price Between 200 AND 500;
- DISTINCT
  - **SELECT** DISTINCT Order_ID
    FROM ORDER_LINE;
- IN Operator
  - SELECT Customer_Name, City, State
    FROM CUSTOMER
    WHERE State IN ('FL','TX','CA','HI');

ORDER BY, GROUP BY

- ORDER BY
  - SELECT Customer_Name,City,State
    FROM CUSTOMER
    WHERE State IN ('FL','TX','CA','HI')
    ORDER BY State, Customer_Name;
- GROUP BY
  - SELECT State, Count(State)
    FROM CUSTOMER
    GROUP BY State;
  - SELECT State,City,Count(City)
    FROM CUSTOMER
    GROUP BY State,City;

GROUP BY

- Write an SQL to get a list of “total quantity of items” for each order
  - SELECT Order_ID, Sum(Quantity)
    FROM ORDER_LINE
    GROUP BY Order_ID;

HAVING

- HAVING is similar to WHERE
- The difference is that WHERE applies to single rows, HAVING applies to Groups
- Example
  - SELECT State, Count(State)
    FROM CUSTOMER
    GROUP BY State
    HAVING Count(State) > 1;

DDL

- CREATE DATABASE
- CREATE TABLE
- DROP TABLE
- ALTER TABLE
- CREATE INDEX
- CREATE VIEW
- DROP VIEW
- CREATE SCHEMA

SQL Database Definition

- Data Definition Language (DDL)
- Major CREATE statements:
  - CREATE SCHEMA – defines a portion of the database owned by a particular user
  - CREATE TABLE – defines a table and its columns
  - CREATE VIEW – defines a logical table from one or more views
- Other CREATE statements: CHARACTER SET, COLLATION, TRANSLATION, ASSERTION, DOMAIN

Table Creation

General syntax for CREATE TABLE

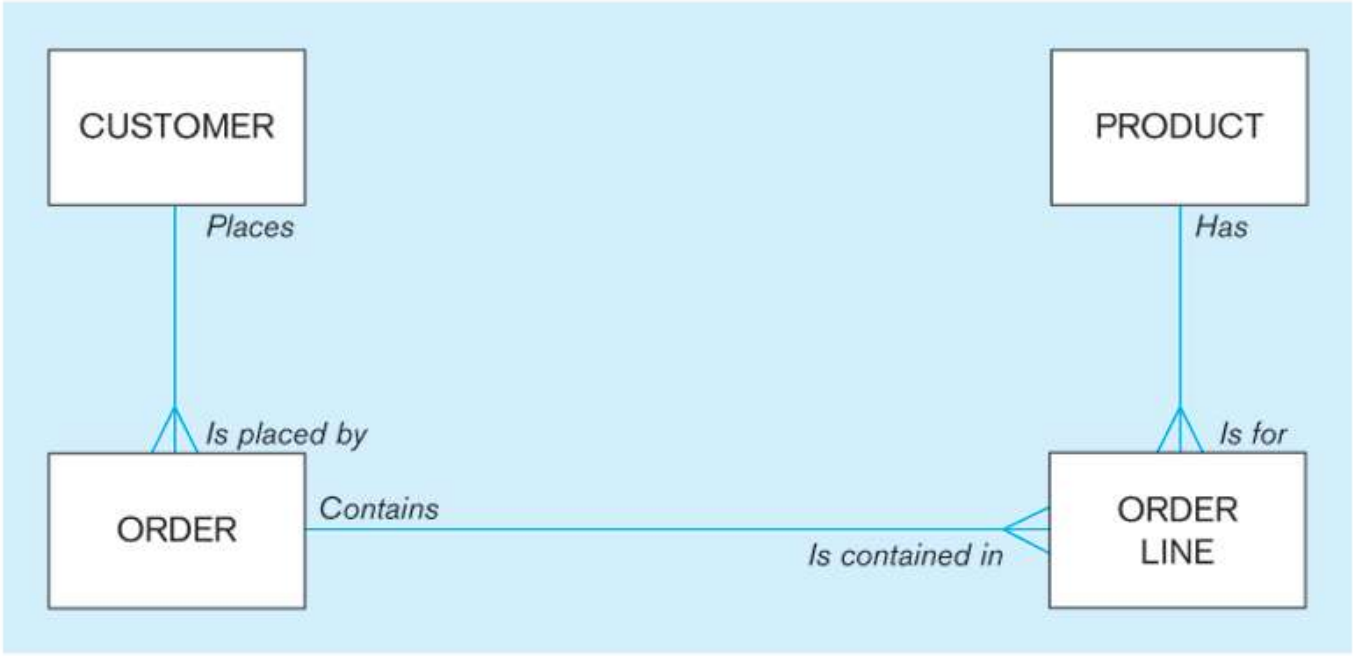
```
CREATE TABLE tablename
( {column definition   [table constraint] } . , . .
[ON COMMIT {DELETE | PRESERVE} ROWS] );

where column definition ::=
column_name
        {domain name | datatype [(size)] }
        [column_constraint_clause . . .]
        [default value]
        [collate clause]

and table constraint ::=
        [CONSTRAINT constraint_name]
        Constraint_type [constraint_attributes]
```

Steps in table creation:

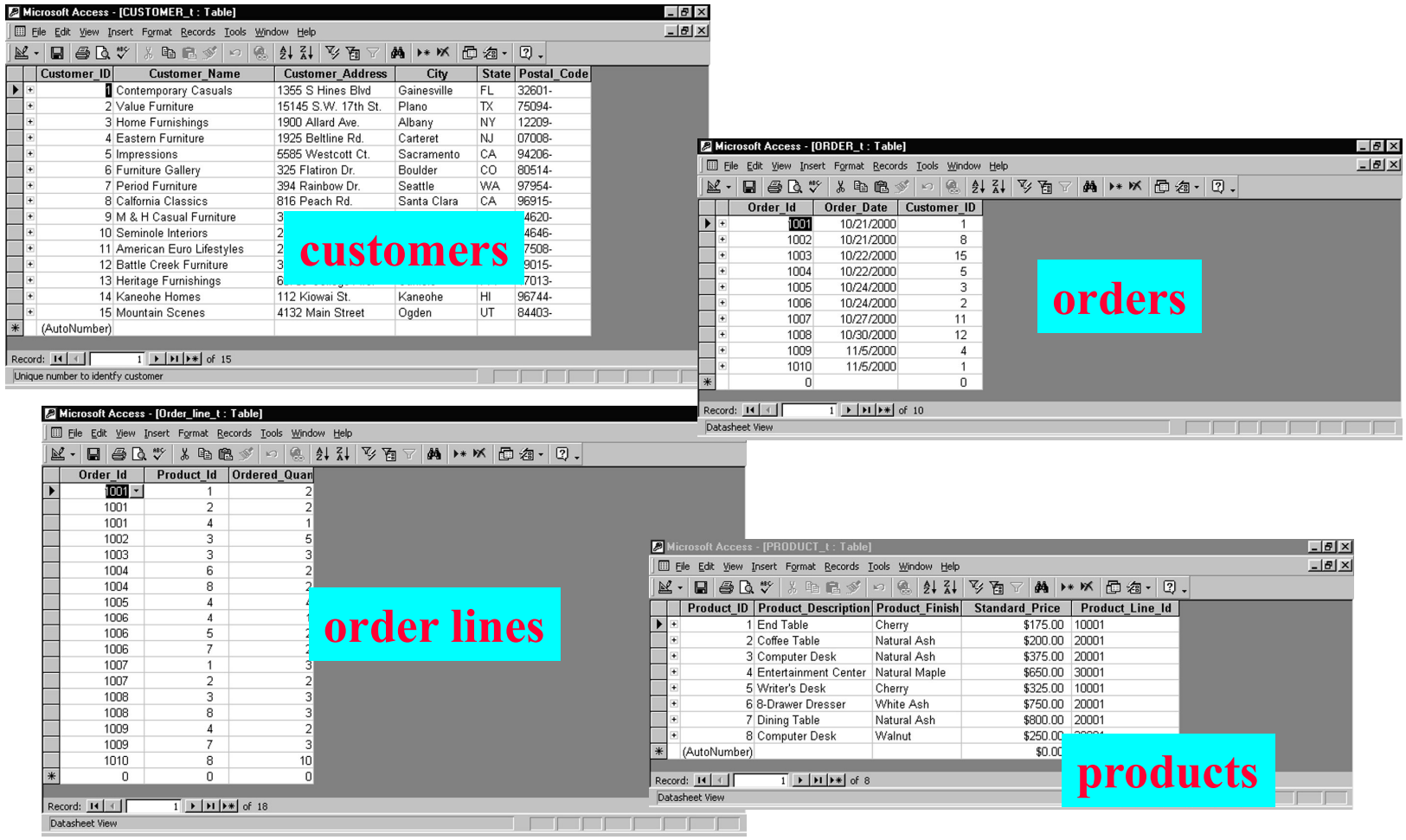
1. Identify data types for attributes
2. Identify columns that can and cannot be null
3. Identify columns that must be unique (candidate keys)
4. Identify primary key-foreign key mates
5. Determine default values
6. Identify constraints on columns (domain specifications)
7. Create the table and associated indexes

The following slides create tables for this enterprise data model

Figure 2-1 Segment from enterprise data model (Pine Valley Furniture Company)



Sample Pine Valley Furniture data



SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
                (CUSTOMER_ID                    NUMBER(11, 0) NOT NULL,
                 CUSTOMER_NAME                  VARCHAR2(25) NOT NULL,
                 CUSTOMER_ADDRESS               VARCHAR2(30),
                 CITY                           VARCHAR2(20),
                 STATE                          VARCHAR2(2),
                 POSTAL_CODE                    VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
                (ORDER_ID                       NUMBER(11, 0) NOT NULL,
                 ORDER_DATE                     DATE          DEFAULT SYSDATE,
                 CUSTOMER_ID                    NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
                (PRODUCT_ID                     INTEGER       NOT NULL,
                 PRODUCT_DESCRIPTION            VARCHAR2(50),
                 PRODUCT_FINISH                 VARCHAR2(20)
                                CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                                          'Red Oak', 'Natural Oak', 'Walnut')),
                 STANDARD_PRICE                 DECIMAL(6,2),
                 PRODUCT_LINE_ID                INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
                (ORDER_ID                       NUMBER(11,0)  NOT NULL,
                 PRODUCT_ID                     NUMBER(11,0)  NOT NULL,
                 ORDERED_QUANTITY               NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
        (CUSTOMER_ID            NUMBER(11, 0) NOT NULL,
         CUSTOMER_NAME          VARCHAR2(25) NOT NULL,
         CUSTOMER_ADDRESS       VARCHAR2(30),
         CITY                   VARCHAR2(20),
         STATE                  VARCHAR2(2),
         POSTAL_CODE            VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
        (ORDER_ID               NUMBER(11, 0) NOT NULL,
         ORDER_DATE             DATE          DEFAULT SYSDATE,
         CUSTOMER_ID            NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
        (PRODUCT_ID             INTEGER       NOT NULL,
         PRODUCT_DESCRIPTION    VARCHAR2(50),
         PRODUCT_FINISH         VARCHAR2(20)
                  CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                    'Red Oak', 'Natural Oak', 'Walnut')),
         STANDARD_PRICE         DECIMAL(6,2),
         PRODUCT_LINE_ID        INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
        (ORDER_ID               NUMBER(11,0)  NOT NULL,
         PRODUCT_ID             NUMBER(11,0)  NOT NULL,
         ORDERED_QUANTITY       NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Defining attributes and their data types

SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
          (CUSTOMER_ID              NUMBER(11, 0) NOT NULL,
           CUSTOMER_NAME            VARCHAR2(25) NOT NULL,
           CUSTOMER_ADDRESS         VARCHAR2(30),
           CITY                     VARCHAR2(20),
           STATE                    VARCHAR2(2),
           POSTAL_CODE              VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
          (ORDER_ID                 NUMBER(11, 0) NOT NULL,
           ORDER_DATE               DATE          DEFAULT SYSDATE,
           CUSTOMER_ID              NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
          (PRODUCT_ID               INTEGER       NOT NULL,
           PRODUCT_DESCRIPTION      VARCHAR2(50),
           PRODUCT_FINISH           VARCHAR2(20)
                    CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                              'Red Oak', 'Natural Oak', 'Walnut')),
           STANDARD_PRICE           DECIMAL(6,2),
           PRODUCT_LINE_ID          INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
          (ORDER_ID                 NUMBER(11,0)  NOT NULL,
           PRODUCT_ID               NUMBER(11,0)  NOT NULL,
           ORDERED_QUANTITY         NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Non-nullable specifications

Note: primary keys should not be null

SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
          (CUSTOMER_ID              NUMBER(11, 0) NOT NULL,
           CUSTOMER_NAME            VARCHAR2(25) NOT NULL,
           CUSTOMER_ADDRESS         VARCHAR2(30),
           CITY                     VARCHAR2(20),
           STATE                    VARCHAR2(2),
           POSTAL_CODE              VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
          (ORDER_ID                 NUMBER(11, 0) NOT NULL,
           ORDER_DATE               DATE          DEFAULT SYSDATE,
           CUSTOMER_ID              NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
          (PRODUCT_ID               INTEGER       NOT NULL,
           PRODUCT_DESCRIPTION      VARCHAR2(50),
           PRODUCT_FINISH           VARCHAR2(20)
                    CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                              'Red Oak', 'Natural Oak', 'Walnut')),
           STANDARD_PRICE           DECIMAL(6,2),
           PRODUCT_LINE_ID          INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
          (ORDER_ID                 NUMBER(11,0)  NOT NULL,
           PRODUCT_ID               NUMBER(11,0)  NOT NULL,
           ORDERED_QUANTITY         NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Identifying primary keys

This is a composite primary key

SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
              (CUSTOMER_ID                NUMBER(11, 0) NOT NULL,
               CUSTOMER_NAME              VARCHAR2(25) NOT NULL,
               CUSTOMER_ADDRESS           VARCHAR2(30),
               CITY                       VARCHAR2(20),
               STATE                      VARCHAR2(2),
               POSTAL_CODE                VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
              (ORDER_ID                   NUMBER(11, 0) NOT NULL,
               ORDER_DATE                 DATE          DEFAULT SYSDATE,
               CUSTOMER_ID                NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));
CREATE TABLE PRODUCT_T
              (PRODUCT_ID                 INTEGER       NOT NULL,
               PRODUCT_DESCRIPTION        VARCHAR2(50),
               PRODUCT_FINISH             VARCHAR2(20)
                          CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                 'Red Oak', 'Natural Oak', 'Walnut')),
               STANDARD_PRICE             DECIMAL(6,2),
               PRODUCT_LINE_ID            INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
              (ORDER_ID                   NUMBER(11,0)  NOT NULL,
               PRODUCT_ID                 NUMBER(11,0)  NOT NULL,
               ORDERED_QUANTITY           NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Identifying foreign keys and establishing relationships

SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
          (CUSTOMER_ID              NUMBER(11, 0) NOT NULL,
           CUSTOMER_NAME            VARCHAR2(25) NOT NULL,
           CUSTOMER_ADDRESS         VARCHAR2(30),
           CITY                     VARCHAR2(20),
           STATE                    VARCHAR2(2),
           POSTAL_CODE              VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
          (ORDER_ID                 NUMBER(11, 0) NOT NULL,
           ORDER_DATE               DATE          DEFAULT SYSDATE,
           CUSTOMER_ID              NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
          (PRODUCT_ID               INTEGER       NOT NULL,
           PRODUCT_DESCRIPTION      VARCHAR2(50),
           PRODUCT_FINISH           VARCHAR2(20)
                CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                          'Red Oak', 'Natural Oak', 'Walnut')),
           STANDARD_PRICE           DECIMAL(6,2),
           PRODUCT_LINE_ID          INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
          (ORDER_ID                 NUMBER(11,0)  NOT NULL,
           PRODUCT_ID               NUMBER(11,0)  NOT NULL,
           ORDERED_QUANTITY         NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Default values and domain constraints

SQL database definition commands for Pine Valley Furniture

```
CREATE TABLE CUSTOMER_T
        (CUSTOMER_ID                    NUMBER(11, 0) NOT NULL,
         CUSTOMER_NAME                  VARCHAR2(25) NOT NULL,
         CUSTOMER_ADDRESS               VARCHAR2(30),
         CITY                           VARCHAR2(20),
         STATE                          VARCHAR2(2),
         POSTAL_CODE                    VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
        (ORDER_ID                       NUMBER(11, 0) NOT NULL,
         ORDER_DATE                     DATE          DEFAULT SYSDATE,
         CUSTOMER_ID                    NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
        (PRODUCT_ID                     INTEGER       NOT NULL,
         PRODUCT_DESCRIPTION            VARCHAR2(50),
         PRODUCT_FINISH                 VARCHAR2(20)
                        CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
         STANDARD_PRICE                 DECIMAL(6,2),
         PRODUCT_LINE_ID                INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
        (ORDER_ID                       NUMBER(11,0)  NOT NULL,
         PRODUCT_ID                     NUMBER(11,0)  NOT NULL,
         ORDERED_QUANTITY               NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

Overall table definitions

Data Integrity Controls

- Referential integrity – constraint that ensures that foreign key values of a table must match primary key values of a related table in 1:M relationships
- Restricting:
  - Deletes of primary records
  - Updates of primary records
  - Inserts of dependent records

Ensuring data integrity through updates

Restricted Update: A customer ID can only be deleted if it is not found in ORDER table.

```
CREATE TABLE CUSTOMER_T
                (CUSTOMER_ID          INTEGER DEFAULT 'C999'   NOT NULL,
                 CUSTOMER_NAME        VARCHAR(40)              NOT NULL,
                 . . .
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID),
ON UPDATE RESTRICT);
```

Cascaded Update: Changing a customer ID in the CUSTOMER table will result in that value changing in the ORDER table to match.

```
. . . ON UPDATE CASCADE);
```

Set Null Update: When a customer ID is changed, any customer ID in the ORDER table that matches the old customer ID is set to NULL.

```
. . . ON UPDATE SET NULL);
```

Set Default Update: When a customer ID is changed, any customer ID in the ORDER tables that matches the old customer ID is set to a predefined default value.

```
. . . ON UPDATE SET DEFAULT);
```

Relational integrity is enforced via the primary-key to foreign-key match

Changing and Removing Tables

- ALTER TABLE statement allows you to change column specifications:
  - ALTER TABLE CUSTOMER_T ADD (TYPE VARCHAR(2))
  - ALTER TABLE Employees
    ADD
    DateOfBirth datetime NULL
    LastRaiseDate datetime NOT NULL
    DEFAULT '2000-01-01'

DROP TABLE statement allows you to remove tables from your schema:

  - DROP TABLE CUSTOMER_T